Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 147 (2016) 912 – 917

Procedia Engineering

www.elsevier.com/locate/procedia

11th conference of the International Sports Engineering Association, ISEA 2016

Activity Recognition in Surfing - A Comparative Study between Hidden Markov Model and Support Vector Machine

Hannes Hoettinger[a], Franziska Mally[a,b,*], Anton Sabo[a,b]

[a] *Institute for Biomedical, Health and Sports Engineering, University of Applied Sciences Technikum Wien, 1200 Vienna, Austria*
[b] *School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, Melbourne, VIC 3083, Australia*

Abstract

The present project describes a comparative study between two different machine learning approaches, the *Hidden Markov Model (HMM)* and *Support Vector Machines (SVMs)*, for activity recognition in surfing, aiming to distinguish surfing from other non-traditional (non-surfing) movements.

The *Hidden Markov Model* has been introduced as a probabilistic or statistical framework for time-varying processes, whereas the *Support Vector Machine* algorithm is probably the most widely used kernel learning algorithm.

Human activities are classified by using only one *Inertial Measurement Unit (IMU)* worn on the chest. A feature set extracted from the raw sensor data is used in the classification process. Feature transformation, in respect of dimensional reduction is implemented with *Principal Component Analysis (PCA)*.

A performance comparison of the classification models is provided in terms of their correct differentiation rates and confusion matrices, as well as their preprocessing and training requirements. 5-fold cross validation is employed to validate the classifiers.

The results indicate that the HMM results in a higher classification accuracy of 91.4% compared to the SVM with an accuracy of 83.4%. The algorithm is capable of classifying time-varying motions from input data of an IMU worn during a surfing session. Moreover, the surfing style between subjects differs widely from left to right waves, right to left waves, goofy or regular footed and the execution itself. However, the implementation of the wave-model allows to train only one data set including every wave data collected and must not separate the data into different forms of execution.

Peer-review under responsibility of the organizing committee of ISEA 2016

Keywords: Hidden Markov Model (HMM), Support Vector Machine (SVM), Inertial Measurement Unit (IMU), Activity Recognition, Principal Component Analysis (PCA)

1. Introduction

Activity recognition formerly was a tedious procedure only possible in limited space and laboratory or instrumented rooms. On-body sensing - originally developed for the entertainment industry - extends its possible application area by observing user's activities in-situ and from their perspective. Such integrated motion sensors are already used for fitness, rehabilitation and sports products offering athlete-feedback on performance and other parameters. [1]

*Corresponding author. Tel.: 43 1 333 40 77 - 383.
E-mail address: franziska.mally@technikum-wien.at

Peer-review under responsibility of the organizing committee of ISEA 2016
doi:10.1016/j.proeng.2016.06.279

The main contribution of this work, however, is suggesting a new application for activity recognition in association with a synchronized video. It provides a comparative study between two different machine learning approaches, the *Hidden Markov Model (HMM)* and *Support Vector Machines (SVMs)*, for activity recognition of non-traditional sports. Specifically in the first instance the focus is on the extreme sport *surfing*, aiming to identify the locations in the measured "session" when wave-riding - as opposed to paddling or waiting for a wave - is executed. According to research only $3.8\% - 5\%$ of the total surfing time corresponds to the wave-riding movements, despite of being the actual surfing goal [2]. The proposed algorithm should automatically detect the locations where the surfer is currently riding a wave.

1.1. Activity Recognition with an Inertial Measurement Unit

An inertial measurement system can only measure relative motion, absolute information like position and orientation have to be calculated by integration, which results in reconstruction errors of the real position or orientation [3]. However, it is also possible to use only the relative change in motion for the movement pattern recognition system, hence avoiding integration and sensor data fusing problems. A defined target motion, riding a wave, should be classified correctly from relative motion changes independent of the position and orientation [1].

2. Data Collection and implementation

The wave data have been collected at the Pacific Ocean (near Valparaiso, CHL) and its surrounding surf spots. No restrictions for the surfing are predefined, to get natural surfing data of all subjects. One IMU (MPU 9150, Invensense Inc., USA) is used for the activity measurements. The raw data is captured with a sampling frequency of 200Hz and a self-built datalogger stores the data on a microSD card. It is placed on the subject's chest in a water-proof case under the wetsuit, with the positive z-axis of the IMU pointing to the front and the positive x-axis to the left. All provided sensor data from the IMU (= 9 axes) are used for the classification algorithm. The skill level of the subjects ranges from intermediate to professional. Figure 1 shows a series of movements executed during the take-off recorded with an installed camera. The model should not classify every single movement, instead the whole series of movements is defined as the target motion. This means, a binary classifier is used, defining only two targets: **wave** or **not-wave**.

With every ridden wave some distinctive movements occur. Figure 1a shows the starting point of an aimed "highlight". A maximum of $5sec$ of paddling is used in the data, then the take-off occurs, where the subject is lifting the upper body (Figure 1b) and stands up on the surfboard (Figure 1c). Finally the subject is riding the wave(Figure 1d). The ending point of the *highlight* is defined as the fall off the board or the exit from the wave, with again $5sec$ of maximum time after this event. Considering this, the complete process of surfing is labeled and used for the training data set. The labeled waves in the data set last from a minimum $5.03sec$ to a maximum of $49.55sec$.

(a) Paddling (b) Push-up (c) Take-off (d) Surfing

Fig. 1. Series of executed movements during surfing = target. motion

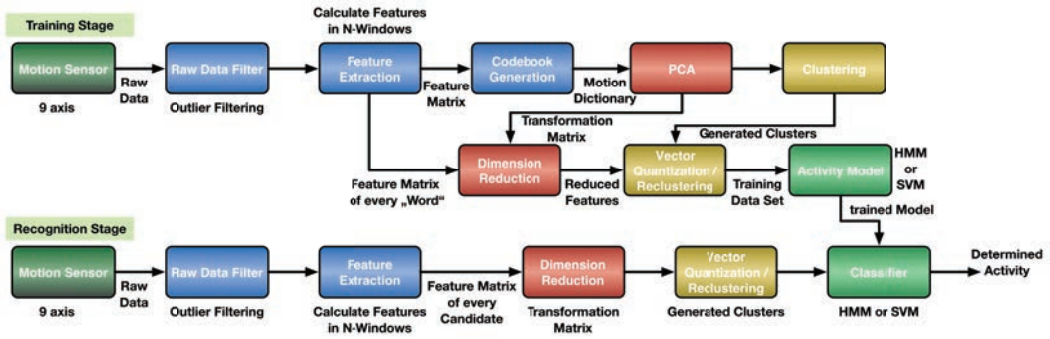


Fig. 2. Basic block flow chart of the algorithm.

Figure 2 describes the basic work-flow of the implemented algorithm - separated into a *training stage* and a *recognition stage*. A "word" is defined as an occurrence, being either a wave or not-wave (IMU data) and candidate as a new occurrence which has to be classified correctly. In the training stage the complete training data set consists of 714 wave occurrences and 1719 not wave occurrences (cross correlated data in the activity-logs). For the cross-validation process, 70% of the training data is used for the training of an activity model and the remaining 30% are used for the recognition stage. The data gets randomly chosen from the complete data set. For the training stage, all extracted features of the complete training data set (including waves and not-waves) are being concatenated to a codebook of the size $n_{features} \times K$ using Principal Component Analysis (PCA) and a clustering method. The number of principal components is defined with $n_{features}$. After this process the defined number of clusters K is assigned to the reduced feature matrix. This is realized using either Gaussian Mixture Models (GMM) or k-means, to finally build the discrete observation codebook, including the assigned cluster centroids in $nfeatures$ dimensions.

For the recognition stage 30% of the complete data set are used and cross-validated to simulate new occurrences. The features are being calculated of every occurrence and dimension reduction is executed using the generated transformation matrix of the training stage. Each frame is clustered into one of the K defined clusters. Afterwards, the classifier (calculated activity model) verifies, which of the two defined motions has been executed (Output = Determined Activity).

The feature extraction and selection stage reduces the signals into features that are discriminative for the activities. Generally speaking, the more clearly each activity can be separated in the feature space, the higher the achieved recognition performance. Two different feature domains are used:

- *Signal-based features:* these are mostly statistical features, such as the mean, variance, or maximum/minimum value. These features are popular due to their simplicity as well as their high performance across a variety of activity recognition problems.
- *Multilevel features:* the data is first clustered,e.g by using k-means or GMM on which occurrence statistics are calculated on a sliding window.

The feature extraction to the signals is applied with overlapped short-time frames. First, the data is blocked into frames of N samples (e.g. $N = 350$). Consecutive frames are spaced ΔN samples apart (e.g. $\Delta N = 50$), such that analysis frame end-times are $m = \{350, 400, ..., N_s\}$. For a typical observation lasting $15s$, the signal length $N_s = 3000$ samples and the number of frames/windows are $T = 54$. The windowing of the signal into the short-time frames is done using a *Hamming* window to reduce the edge effects. Linear Predictive Coding (LPC) coefficients are computed by using the Levinson-Durbin recursion, and then converted to Q cepstral coefficients [4,5]. To add dynamic information, Q differenced cepstral coefficients are also computed. These features are computed for each given short-time frame, including also statistical features (signal-based features with *max, min, variance, root mean square, mean absolute-, integrated absolute value and waveform length*). Furthermore, a Fast Fourier Transform for spectral analysis is used to sort the occurring amplitudes descendant, to only consider the first highest amplitude values for the feature vector. The feature vector is then constructed by concatenating all the feature vectors for each of the IMU axes.

3. Hidden Markov Model

Prior to training any of the HMMs for the individual utterance, a set of (continuous) observation vectors from a large corpus of motion is used to derive the so-called codebook. Subsequently, any observation vector used for either training or recognition is quantized using this codebook. It is sufficient to assign an observation to a single integer, say k, with $1 \leq k \leq K$. However, HMM training is not fixed to a certain input space size, since the *forward-backward algorithm* considers each observation on its own compared to SVMs.

The HMM algorithm is based on the fundamentals of the proposed algorithm of ITU-Copenhagen[5]. It implements an ergodic-discrete-HMM, which means, every state can be reached from any state in a finite number of steps. For each word, the (quantized) observation sequences are used to train the HMM for the specific word, using the F-B reestimation algorithm with an initial random model. The algorithm is able to handle time-varying data, as the input is a sequence of assigned clusters, not forced to have a certain length. The output arguments of the F-B reestimation are the S-by-S state transition matrix A (S...number of states), the K-by-S observation probability matrix B, and the initial state probability vector π, building the HMM-model for the defined word. This is executed two times, to build a HMM-model for the wave data and a second model for the not-wave data, each defined with the output of the reestimation algorithm $\lambda_{1,2} = (A, B, \pi)$. The probability measures for the two defined HMMs of a new sequence, given by A, B, and π, are then used to calculate the log-likelihoods for the HMMs - $P_1(O|\lambda_1)$ and $P_2(O|\lambda_2)$. The word associated with the HMM of highest log-likelihood is declared to be the recognized word.

4. Support Vector Machine

The SVM basically implements the following idea: It maps the input vector x (training data set) into a high-dimensional feature space $\mathcal{H}$ through some nonlinear mapping, using the Kernel-trick. In this space, an optimal separating hyperplane is constructed. In most cases, SVM generalization performance either matches or is significantly better than that of competing methods in supervised learning [6].

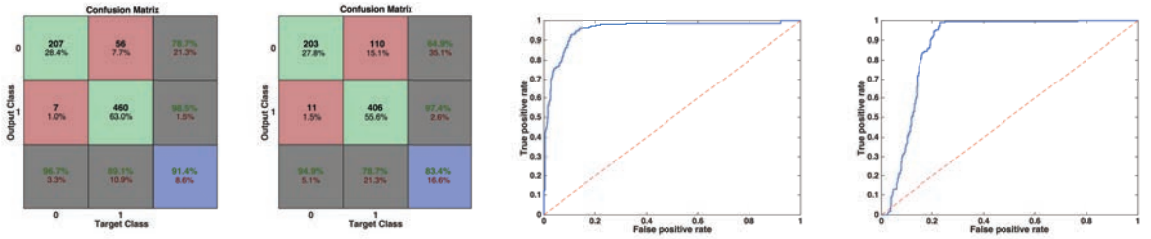
A library for SVMs (LIBSVM toolbox) is used in the MATLAB environment [7]. The preprocessing, feature extraction with dimensional reduction and codebook generation are identical to the HMM-process. However, the SVM has one disadvantage compared to HMM in the preparation of the observation vector for the training and recognition. The SVM algorithm can only deal with observation sequences of the same length and cannot handle time-varying data at first instance. This means the training data matrix for the SVM must contain sequences with a defined length. The histogram of the assigned clusters is used to build the training/recognition sequence for the classifier. For the training of the SVM each observation needs a label, either $+1$ = wave or -1 = not-wave. Furthermore another "clustering" approach for the SVM algorithm is introduced. As the features are reduced to a very low dimension, for example 2, instead of GMM or k-means, the histogram of the features is used to build the training sequence. With this procedure the SVM requirement - that the training data matrix must contain sequences with a defined length - is again satisfied.

5. Experimental results

The confusion matrices in Figure 3 are based on a Monte Carlo analysis to determine the best parameters for each model. The training and recognition is repeated five times such that 30% training occurrences are randomly selected from the whole data set as the validation data. The five classification rates are averaged to produce a single estimate of the performance. Looking at the confusion matrices of the different techniques, the Hidden Markov Model shows a 8% higher accuracy on the same data set as the Support Vector Machine. The HMM-model performs with an accuracy of 91.4%, whereas the SVM only is capable of classifying 83.4% correctly.

Type I error of the SVM confusion matrix is 110, compared to HMM with 56, while the Type II error is almost identical. The SVM generalizes not well, which means, it is very likely that a not-wave occurrence will be classified as a wave using the SVM.

Varying the decision threshold over an interval of the activity-models, a set of true positive rates (TPRs) and the corresponding false positive rates (FPRs) are obtained and plotted as a receiver operating characteristic. A test with perfect discrimination has a ROC curve that passes through the upper left corner (100% sensitivity = TPR, 100%



(a) HMM Result (b) SVM Result (c) HMM (d) SVM

Fig. 3. Confusion Matrix of the cross-validation for (a) HMM and (b) SVM; class 0 = wave, class 1 = not-wave; Receiver Operating Characteristic (ROC) curve for the (c) Hidden Markov Model and (d) Support Vector Machine classifier

specificity = FPR). HMM has a higher area under curve (AUC) measure for the classification. The results suggest that HMM has considerably better in-sample average performance for the given data set.

To finally test the algorithm, two new activity-logs from two different surfers are used. The test is based on the trained HMM-model, as it shows the best classification result. Furthermore, a cross-correlation algorithm to provide the candidates for the recognition stage is included. This should showcase the behavior of the final algorithm to detect the defined motion, riding a wave, in a given activity-log automatically. The first activity-log lasts 1h24min (Surfspot: Ritoque, Chile) and the second 1h45min (Surfspot: Renaca, Chile). Looking at the recorded video of the session, the activity-logs include different kinds of motion, like running towards the ocean, paddling, sitting, turning, etc. As the activity-logs are not labeled, the classification result has to be checked with the synchronized video material. Table 1 shows that every wave is found by the cross-correlation and also correctly classified by the HMM. The surf session in Renaca led to 2 Type I Errors. The processing time $< 5sec$ is remarkably low for a surfing session of 1h45min at $200Hz$ sampling rate.

Table 1. Results of the combined algorithm (Time-series cross-correlation and HMM-model) for two new activity-logs

Activity log	Candidates	Processing Time	Waves	Type I Error	Type II Error
Ritoque	21	4.19*sec*	12	0	0
Renaca	50	4.64*sec*	21	2	0

6. Discussion

The proposed algorithm is able to classify time-varying motions from input data of an IMU based on machine learning techniques by using feature extraction, dimensional reduction and clustering methods. Moreover, the wave-riding style between subjects differs widely from left to right waves, right to left waves, goofy or regular footed and the execution itself. The implementation of the wave-model allows to train only one data set including every wave data collected and does not have to separate the data into different forms of execution. The best result of 91.4% accuracy is achieved with the HMM, by using only one IMU mounted on the chest.

The confusion matrices in Figure 3 show, that the overall cross-validated performance of the ergodic-discrete-HMM using k-means is superior to the SVMs. Out of a data set of 214 waves only 7 are not classified correctly. Furthermore the Type I Error is twice as big for the SVM. Type II Error is set to a high priority in the random parameter search, as the algorithm should find every wave in the activity-log to ensure that the user is able to get every ridden wave of the session. Type I Error equates with adding undesired motions from the activity-log to the result of detected waves.

The HMM shows very good performance for time-varying data. As already mentioned the SVM has the disadvantage that the algorithm can only deal with observation sequences of the same length and cannot handle time-varying data at first instance. The F-B reestimation algorithm (HMM), however, is able to deal with observation sequences of different length and is able to generate good probability models for the given problem of classifying wave-riding

motions. Moreover it has been proved that the good classification properties of HMM for speech recognition have been successfully adapted to motion recognition.

The SVM performed best with a linear kernel and the proposed histogram method. It seems that the SVM is not able to split the data in high-dimensional space accurately, which indicates a bad linear regression and a bad generalization. The HMM result for time-series data is 8% higher. The sequential data is converted to state changes, which seems to better express different executed motions measured with an IMU. By a probability approach it is possible to distinguish between trained motions. A disadvantage of the HMM algorithm is the random initial guess of $\lambda = (A, B, \pi)$ in the F-B reestimation algorithm. The random guess leads to a local optimum in the training stage. This is proved by using the same data set twice and the HMM displays different classification results.

As the hardware only consists of an IMU and some peripheral components to process and store the data, it is a low-cost device, despite with 91.4% accuracy to spot the highlights during a surf session. Since IMUs are small-built devices, it would be possible to fit the sensor into an action-camera. Based on machine learning the feature of automatically detecting specific highlights/motions in a video could be added.

6.1. Unbalanced training data set

Geometrically, the SVM modeling algorithm works by constructing a separating hyperplane with the maximal margin. However, a SVM classifier can be sensitive to high class imbalance, resulting in a drop in the classification performance on the positive class. The algorithm tends to generate a classifier that has a strong estimation bias towards the majority class, resulting in a large number of false negatives. This behavior is observed when using the same data set as for the HMM training. Therefore only 500 wave and 500 not-wave occurrences have been used for the SVM training. There are three general solution types to this problem, the cost-sensitive learning, oversampling the minority class or undersampling the majority class.

For a highly imbalanced data set, there may be many redundant or noisy negative samples. Random undersampling is a common undersampling approach for re-balancing the data set to achieve better data distribution, which is being implemented. However, random undersampling suffers from information loss [8].

This information loss, could lead to a worse classification result shown in Figure 3 compared to HMM. The implementation of the HMM reestimation algorithm is capable of dealing with unbalanced data sets to build a probability model. The SVM considers the data set of wave and not-wave occurrences in its entirety to fit the separating hyperplane, whereas the HMM builds two probability models separately.

Since the HMM has provided very good results with the imbalanced data set further investigation on this problem was not initiated.

References

[1] A. Bulling, U. Blanke, B. Schiele, A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors, ACM Computing Surveys 46 (2014).

[2] D. D. Bona, M. A. Marques, M. V. Correia, Instrumentation of a surfboard to evaluate surfing performance, International Conference on Remote Engineering and Virtual Instrumentation (REV) 11 (2014) 339–343.

[3] L. Klingbeil, Entwicklung eines modularen und skalierbaren Sensorsystems zur Erfassung von Position und Orientierung bewegter Objekte, Dissertation, Bonn - Physikalisches Institut, 2006.

[4] J. R. J. Deller, J. H. L. Hansen, J. G. Proakis, Discrete-Time Processing of Speech Signals (IEEE Press Classic Reissue), IEEE Signal Processing Society, 1999.

[5] ITU-Copenhagen, Speech Recognition: A library for hidden markov models, Matlab Functions (2000). Source available at `http://www.itu.dk/courses/TKG/E2005/exercises.html`.

[6] C. J. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Disovery (1998) 121–167.

[7] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[8] T. Yuchun, Z. Yan-Qing, N. Chawla, S. Krasser, SVMs Modeling for Highly Imbalanced Classification, IEEE Transactions on Systems Man and Cybernetics Part B 39 (2009) 281–288.